Task 1 Exploratory Data Analysis

subtask:

- Check and clean the dataset,gathering overall insights about the data
- Removed the outlier and segment the data by transaction date and time and visulise the data
- Draw some insights about the location information

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
pd.set_option('display.max_columns',100)
import warnings
warnings.filterwarnings("ignore")
```

Load the data

In [2]:
```
df=pd.read_excel('ANZ synthesised transaction dataset.xlsx')
print(df.shape)
df.head()
```

```
(12043, 23)
```

Out[2]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_descr
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES

Basic checks

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 12043 entries, 0 to 12042
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   status             12043 non-null  object
 1   card_present_flag  7717 non-null   float64
 2   bpay_biller_code   885 non-null    object
 3   account            12043 non-null  object
 4   currency           12043 non-null  object
 5   long_lat           12043 non-null  object
 6   txn_description    12043 non-null  object
 7   merchant_id        7717 non-null   object
 8   merchant_code      883 non-null    float64
 9   first_name         12043 non-null  object
 10  balance            12043 non-null  float64
 11  date               12043 non-null  datetime64[ns]
 12  gender             12043 non-null  object
 13  age                12043 non-null  int64
 14  merchant_suburb    7717 non-null   object
 15  merchant_state     7717 non-null   object
 16  extraction         12043 non-null  object
 17  amount             12043 non-null  float64
 18  transaction_id     12043 non-null  object
 19  country            12043 non-null  object
 20  customer_id        12043 non-null  object
 21  merchant_long_lat  7717 non-null   object
 22  movement           12043 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(1), object(17)
memory usage: 2.1+ MB
```

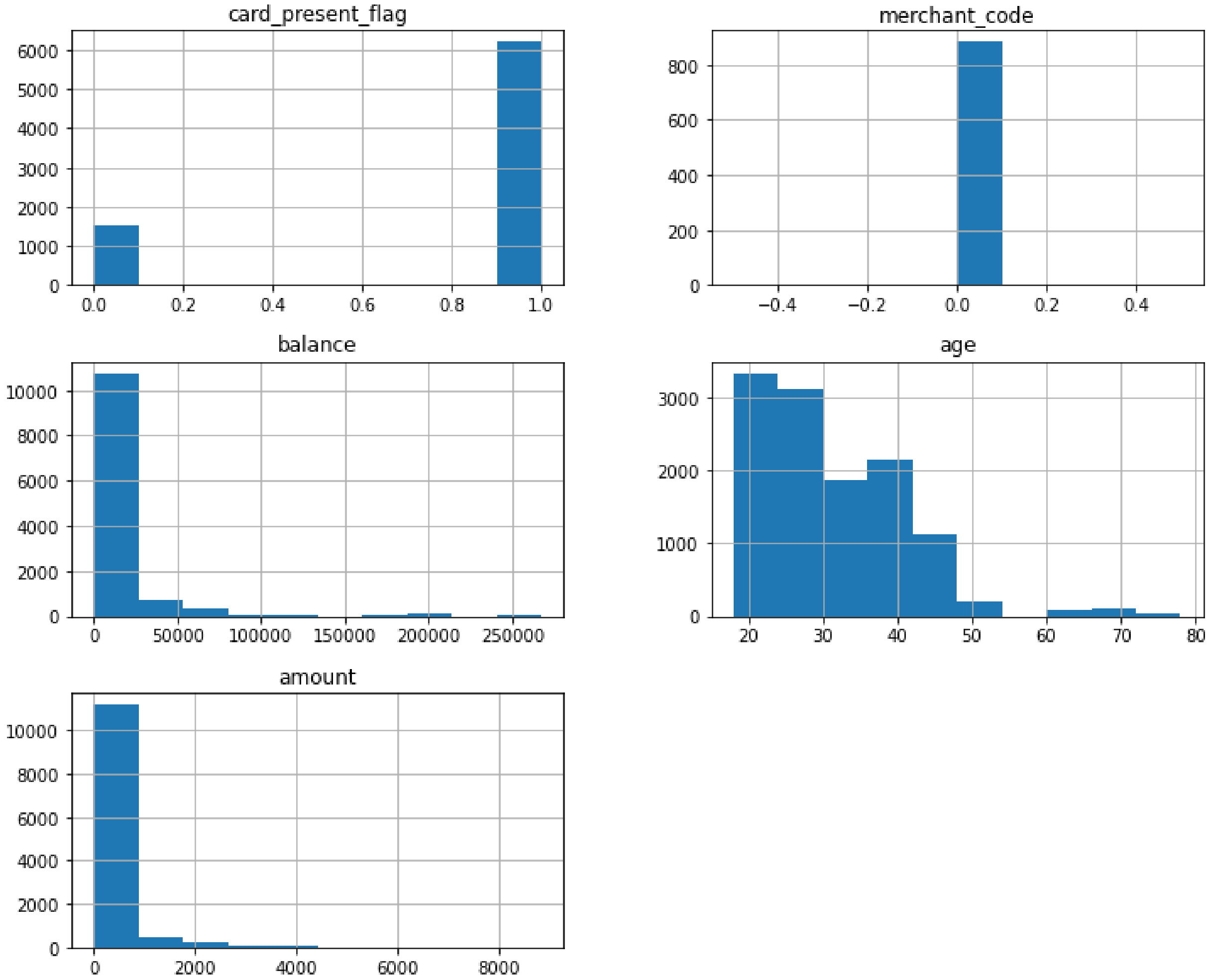
Through the output, we can see that the card_presen_flag, merchant_id,merchant_suburb,merchant_state,merchant_long_lat have the same numbers of null value, and the bpay_biller_code and merchant_code have null value over 90%, we will deal this missing data later

In [4]:
```
df.describe()
```

Out[4]:

	card_present_flag	merchant_code	balance	age	amount
count	7717.000000	883.0	12043.000000	12043.000000	12043.000000
mean	0.802644	0.0	14704.195553	30.582330	187.933588
std	0.398029	0.0	31503.722652	10.046343	592.599934
min	0.000000	0.0	0.240000	18.000000	0.100000
25%	1.000000	0.0	3158.585000	22.000000	16.000000
50%	1.000000	0.0	6432.010000	28.000000	29.000000
75%	1.000000	0.0	12465.945000	38.000000	53.655000
max	1.000000	0.0	267128.520000	78.000000	8835.980000

In [5]:
```
df.hist(figsize=(12,10));
```



We can see that all the merchant_code are zero, although it is not null.The card_present_flag either 0 or 1, it should be a category data.As for the distribution of amount, balance and age are highly left skewed, we will analysis them in the next step

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
df.bpay_biller_code.value_counts()
```

Out[7]:
```
0                                        883
 LAND WATER & PLANNING East Melbourne      1
 THE DISCOUNT CHEMIST GROUP                1
Name: bpay_biller_code, dtype: int64
```

In [8]:
```
df.nunique()
```

Out[8]:
```
status                   2
card_present_flag        2
bpay_biller_code         3
account                100
currency                 1
long_lat               100
txn_description          6
merchant_id           5725
merchant_code            1
first_name              80
balance              12006
date                    91
gender                   2
age                     33
merchant_suburb       1609
```

```
merchant_state           8
extraction            9442
amount                4457
transaction_id       12043
country                  1
customer_id            100
merchant_long_lat     2703
movement                 2
dtype: int64
```

In [9]:
```
df.value_counts
```

Out[9]:
```
<bound method DataFrame.value_counts of            status  card_present_flag b
pay_biller_code        account  \
0      authorized                1.0              NaN  ACC-1598451071
1      authorized                0.0              NaN  ACC-1598451071
2      authorized                1.0              NaN  ACC-1222300524
3      authorized                1.0              NaN  ACC-1037050564
4      authorized                1.0              NaN  ACC-1598451071
...         ...                  ...              ...             ...
12038  authorized                0.0              NaN  ACC-3021093232
12039  authorized                1.0              NaN  ACC-1608363396
12040  authorized                1.0              NaN  ACC-3827517394
12041  authorized                1.0              NaN  ACC-2920611728
12042  authorized                1.0              NaN  ACC-1443681913

      currency       long_lat txn_description  \
0          AUD  153.41 -27.95             POS
1          AUD  153.41 -27.95       SALES-POS
2          AUD  151.23 -33.94             POS
3          AUD  153.10 -27.66       SALES-POS
4          AUD  153.41 -27.95       SALES-POS
...        ...            ...             ...
12038      AUD  149.83 -29.47             POS
12039      AUD  151.22 -33.87       SALES-POS
12040      AUD  151.12 -33.89             POS
12041      AUD  144.96 -37.76       SALES-POS
12042      AUD  150.92 -33.77       SALES-POS

                                merchant_id  merchant_code first_name  \
0      81c48296-73be-44a7-befa-d053f48ce7cd            NaN      Diana
1      830a451c-316e-4a6a-bf25-e37caedca49e            NaN      Diana
2      835c231d-8cdf-4e96-859d-e9d571760cf0            NaN    Michael
3      48514682-c78a-4a88-b0da-2d6302e64673            NaN     Rhonda
4      b4e02c10-0852-4273-b8fd-7b3395e32eb0            NaN      Diana
...                                     ...            ...        ...
12038  32aa73dc-b7c2-4161-b14d-6271b96ce792            NaN    Melissa
12039  296a0500-8552-48ac-ac81-ec37065b568e            NaN     Robert
12040  e5975ab4-08f7-4725-a369-24cc0e35ed6e            NaN      Craig
12041  af49051a-591d-4b08-bd3c-27730b70ed37            NaN      Tyler
12042  f31f4b14-2040-40ec-a120-b141bb274cbd            NaN       Ryan

        balance       date gender  age merchant_suburb merchant_state  \
0         35.39 2018-08-01      F   26         Ashmore            QLD
1         21.20 2018-08-01      F   26          Sydney            NSW
2          5.71 2018-08-01      M   38          Sydney            NSW
3       2117.22 2018-08-01      F   40         Buderim            QLD
4         17.95 2018-08-01      F   26   Mermaid Beach            QLD
...         ...        ...    ...  ...             ...            ...
12038  14054.14 2018-10-31      F   30        Ringwood            VIC
12039   9137.79 2018-10-31      M   20          Casula            NSW
12040  45394.57 2018-10-31      M   28      Kings Park            NSW
12041  11350.67 2018-10-31      M   69        Oakleigh            VIC
12042   5517.91 2018-10-31      M   31          Mascot            NSW

                     extraction  amount                    transaction_id
\
0      2018-08-01T01:01:15.000+0000   16.25  a623070bfead4541a6b0fff8a09e706c
1      2018-08-01T01:13:45.000+0000   14.19  13270a2a902145da9db4c951e04b51b9
```

```
2       2018-08-01T01:26:15.000+0000    6.42  feb79e7ecd7048a5a36ec889d1a94270
3       2018-08-01T01:38:45.000+0000   40.90  2698170da3704fd981b15e64a006079e
4       2018-08-01T01:51:15.000+0000    3.25  329adf79878c4cf0aeb4188b4691c266
...                              ...     ...                               ...
12038   2018-10-31T23:09:06.000+0000    9.79  f2e3e695c2ee4c50a4c8747f852cbe2e
12039   2018-10-31T23:21:46.000+0000   63.87  56e147e5485f4683b9076fcaaed76640
12040   2018-10-31T23:34:25.000+0000   43.96  2fdd4681827343f6af2e6519644a684a
12041   2018-10-31T23:47:05.000+0000   30.77  74aa9cd7e4af4c6d9cd7dbd28e9aedc9
12042   2018-10-31T23:59:44.000+0000   22.36  6d5218e04e8040b9996850ce11a19426

          country     customer_id merchant_long_lat movement
0       Australia  CUS-2487424745     153.38 -27.99    debit
1       Australia  CUS-2487424745     151.21 -33.87    debit
2       Australia  CUS-2142601169     151.21 -33.87    debit
3       Australia  CUS-1614226872     153.05 -26.68    debit
4       Australia  CUS-2487424745     153.44 -28.06    debit
...           ...             ...               ...      ...
12038   Australia    CUS-55310383     145.23 -37.81    debit
12039   Australia  CUS-2688605418     150.88 -33.96    debit
12040   Australia  CUS-2663907001     150.92 -33.74    debit
12041   Australia  CUS-1388323263     145.09 -37.91    debit
12042   Australia  CUS-3129499595     151.19 -33.93    debit

[12043 rows x 23 columns]>
```

The column of currency and country not provide valid information, we will clean it in the clean data step later. For the bpay biller code, it just have 2 valid value which is definitely not enough.

Since it is a synthesised transaction data containing 3 months' transactions for 100 customers,we need to check if there any date or any customer is missing

In [10]:

```python
print('There have', df['customer_id'].nunique(),'unique Customer ID')
print('There have', df['transaction_id'].nunique(),'unique Customer ID')
print('There have', df['date'].nunique(),'unique date')
#See which value are present in particular column
print("Customer ID:\n", df['customer_id'].value_counts(),"\n")
print("Date:\n", df['date'].value_counts(),"\n")
print("Date:\n", df['date'].sort_values,"\n")
```

```
There have 100 unique Customer ID
There have 12043 unique Customer ID
There have 91 unique date
Customer ID:
 CUS-2487424745    578
CUS-2142601169    303
CUS-3026014945    292
CUS-3378712515    260
CUS-1614226872    259
                 ...
CUS-3395687666     40
CUS-3201519139     37
CUS-1646183815     34
CUS-495599312      31
CUS-1739931018     25
Name: customer_id, Length: 100, dtype: int64

Date:
 2018-09-28    174
2018-08-17    172
2018-10-05    168
2018-10-17    162
2018-09-14    161
             ...
2018-08-06     99
2018-08-20     97
2018-10-23     96
2018-10-08     95
```

```
2018-10-30     89
Name: date, Length: 91, dtype: int64

Date:
 <bound method Series.sort_values of 0       2018-08-01
1        2018-08-01
2        2018-08-01
3        2018-08-01
4        2018-08-01
            ...
12038    2018-10-31
12039    2018-10-31
12040    2018-10-31
12041    2018-10-31
12042    2018-10-31
Name: date, Length: 12043, dtype: datetime64[ns]>
```

There do have 100 unique customer id and 12043 unique transaction. However, 3 months between 2018-08-01 to 2018-10-31 should have 92 days but there only have 91, one day's data was missing.

In [11]:
```
pd.date_range(start='2018-08-01',end='2018-10-31').difference(df.date)
```

Out[11]:
```
DatetimeIndex(['2018-08-16'], dtype='datetime64[ns]', freq=None)
```

The missing data was 2018-08-16.

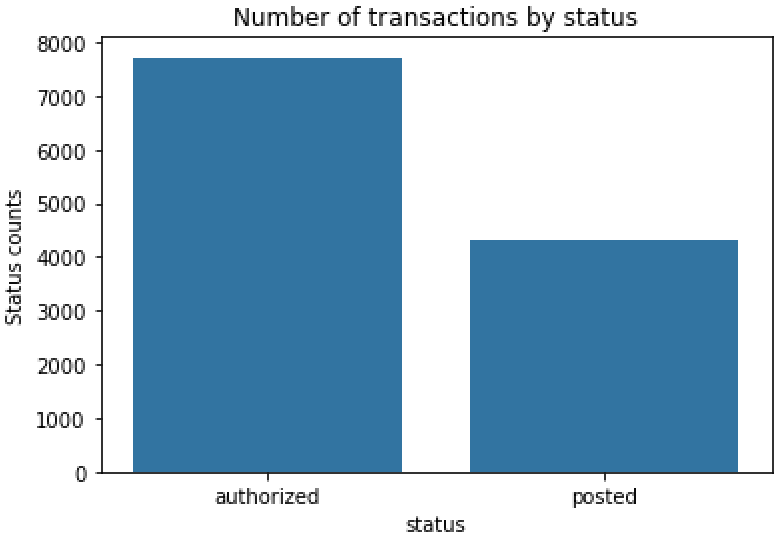
Exploratory Data Analysis

Categorical variables

The categories we gonna to analysis are following, since other variables we already found they won't provide us with much information:

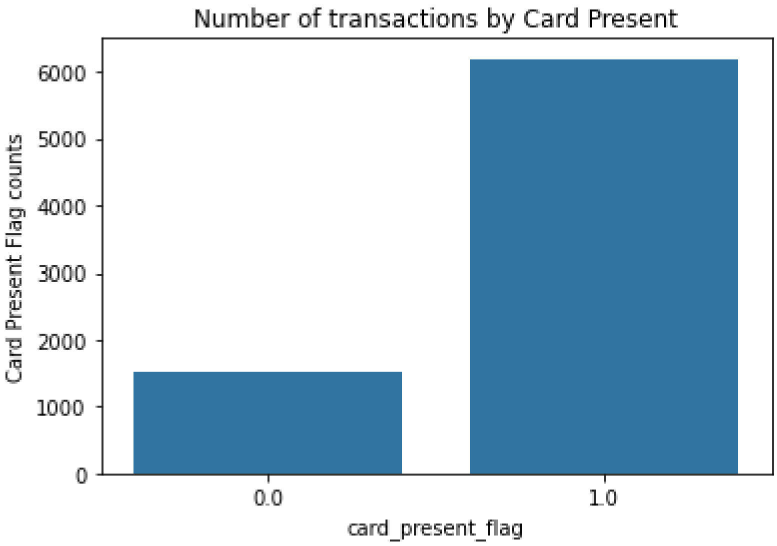
- status
- card_present_flag
- long_lat
- txn_description
- gender
- age
- merchant_suburb
- merchant_state
- extraction
- merchant_long_lat
- movement

In [12]:
```
# Check the 2 status distribution
plt.figure(figsize=(6,4))
base_color=sns.color_palette()[0]
sns.countplot(data=df,x='status',color=base_color)
plt.ylabel("Status counts");
plt.title('Number of transactions by status');
```



Most transactions were authorized which means they already been approved, and other trasactions still in the process

In [13]:
```
plt.figure(figsize=(6,4))
base_color=sns.color_palette()[0]
sns.countplot(data=df,x='card_present_flag',color=base_color)
plt.ylabel("Card Present Flag counts");
plt.title('Number of transactions by status');
plt.ylabel("Card Present Flag counts");
plt.title('Number of transactions by Card Present');
```



A transaction is only considered to be “card present” if payment details are captured in person, at the time of the sale. Majority transactions are card present

In [14]:
```
#Long_lat
print(df['long_lat'].value_counts())
df.long_lat.head()
```

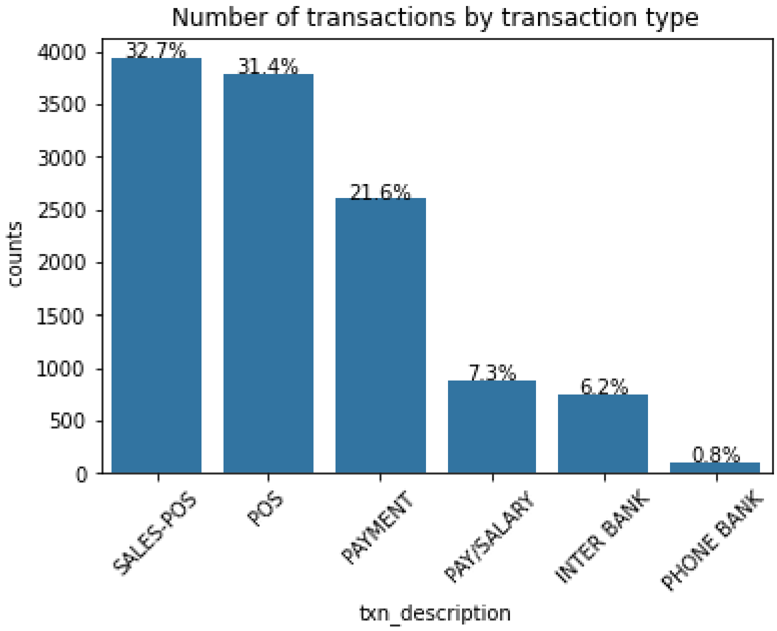
```
153.41 -27.95    578
151.23 -33.94    303
116.06 -32.00    292
145.45 -37.74    260
153.10 -27.66    259
                ...
149.03 -35.25     40
149.19 -21.15     37
145.09 -37.82     34
130.98 -12.49     31
```

```
147.61 -37.82     25
Name: long_lat, Length: 100, dtype: int64
```

Out[14]:
```
0    153.41 -27.95
1    153.41 -27.95
2    151.23 -33.94
3    153.10 -27.66
4    153.41 -27.95
Name: long_lat, dtype: object
```

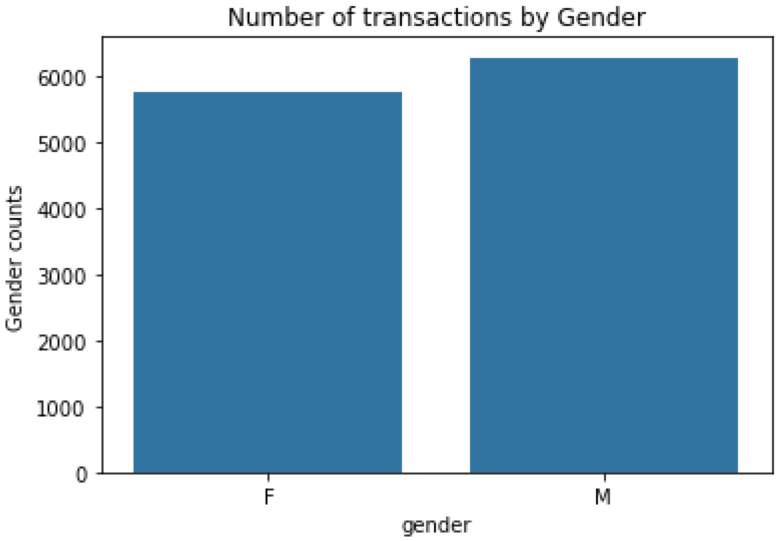
These are the coordinates where the transactions were made, we will explore it deep in further.

In [15]:
```
# txn_description
plt.figure(figsize=(6,4))
base_color=sns.color_palette()[0]
n_txn_description=df['txn_description'].value_counts().sum()
txn_description_counts=df['txn_description'].value_counts()
txn_description_order=txn_description_counts.index
sns.countplot(data=df,x='txn_description',order=txn_description_order,color=b
locs,labels=plt.xticks(rotation=45)
n_txn_description=df['txn_description'].value_counts().sum()
txn_description_counts=df['txn_description'].value_counts()
txn_description_order=txn_description_counts.index
for loc, label in zip(locs,labels):
    count=txn_description_counts[label.get_text()]
    plt_string='{:0.1f}%'.format(100*count/n_txn_description)
    plt.text(loc,count+2,plt_string,ha='center',color='black')
plt.ylabel(" counts");
plt.title('Number of transactions by transaction type');
```



These are the transaction types,mostly transaction are POS,this also can explain that merchant columns has missing values, since not all transactions have merchants.

In [16]:
```
#gender
plt.figure(figsize=(6,4))
base_color=sns.color_palette()[0]
sns.countplot(data=df,x='gender',color=base_color)
plt.ylabel("Gender counts");
plt.title('Number of transactions by Gender');
```



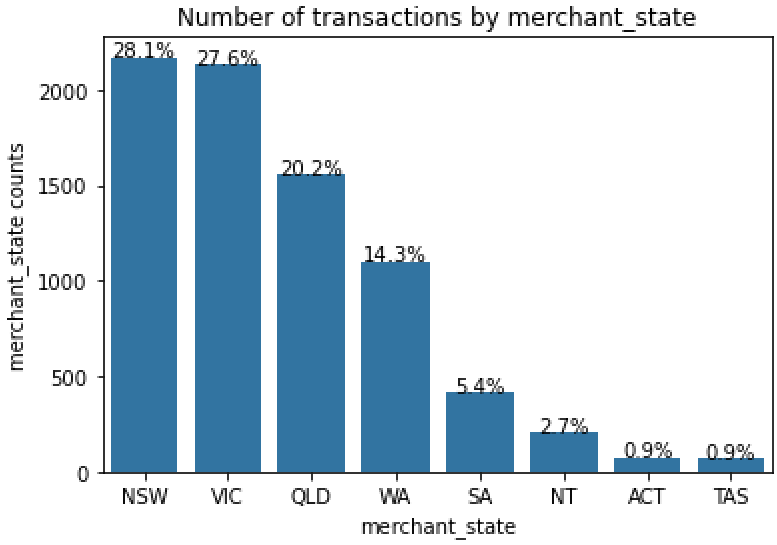
There are more male customer that female

In [17]:
```
# Merchant suburb
df.merchant_suburb.value_counts()
```

Out[17]:
```
Melbourne           255
Sydney              233
Southport            82
Brisbane City        79
Chatswood            55
                   ...
Fairfield Heights     1
Yarra Glen            1
Riverhills            1
West Mackay           1
Cowra                 1
Name: merchant_suburb, Length: 1609, dtype: int64
```

There are subrubs where the Mercant transaction made

In [18]:
```
#Merchant state
plt.figure(figsize=(6,4))
base_color=sns.color_palette()[0]
n_merchant_state=df['merchant_state'].value_counts().sum()
merchant_state_counts=df['merchant_state'].value_counts()
merchant_state_order=merchant_state_counts.index
sns.countplot(data=df,x='merchant_state',color=base_color,order=merchant_stat
locs,labels=plt.xticks()
for loc, label in zip(locs,labels):
    count=merchant_state_counts[label.get_text()]
    plt_string='{:0.1f}%'.format(100*count/n_merchant_state)
    plt.text(loc,count+2,plt_string,ha='center',color='black')
plt.ylabel("merchant_state counts");
plt.title('Number of transactions by merchant_state');
```



There are states where the mercant transaction made, the top2 state are NSW and VIC, the ACT and TAS has the least transactions

In [19]:
```
#Extraction
df.extraction.head()
```

Out[19]:
```
0    2018-08-01T01:01:15.000+0000
1    2018-08-01T01:13:45.000+0000
2    2018-08-01T01:26:15.000+0000
3    2018-08-01T01:38:45.000+0000
4    2018-08-01T01:51:15.000+0000
Name: extraction, dtype: object
```

These are the timestamps for each transaction, since we already have a date column, we can delete the date component out of the extraction column.

In [20]:
```
#merchant_long_lat
print(df['merchant_long_lat'].value_counts())
df.merchant_long_lat.head()
```

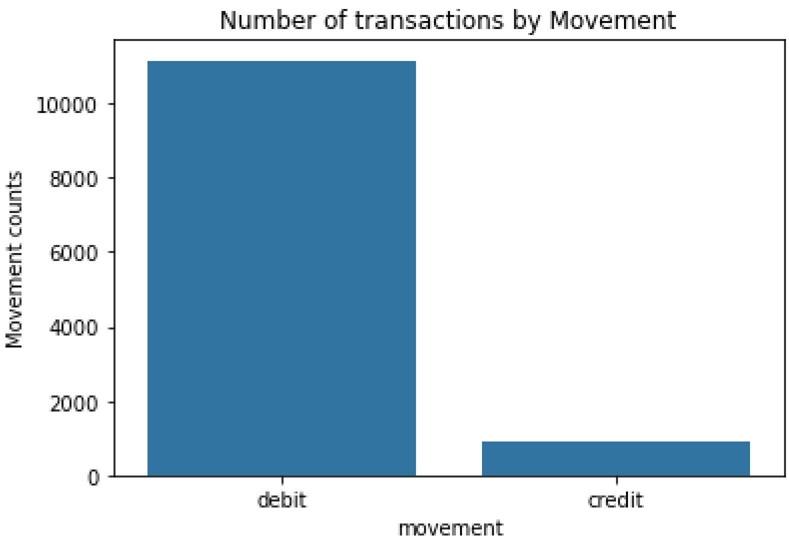
```
151.21 -33.87    145
144.96 -37.82     85
144.97 -37.81     59
144.96 -37.81     56
153.02 -27.47     46
                ...
115.73 -33.03      1
144.86 -37.81      1
152.97 -27.41      1
145.04 -37.9       1
153.41 -28.1       1
Name: merchant_long_lat, Length: 2703, dtype: int64
```

Out[20]:
```
0    153.38 -27.99
1    151.21 -33.87
2    151.21 -33.87
3    153.05 -26.68
4    153.44 -28.06
Name: merchant_long_lat, dtype: object
```

These are the coordinates where the transactions were made, we will explore it deep in further.

In [21]:
```
#movement
plt.figure(figsize=(6,4))
base_color=sns.color_palette()[0]
sns.countplot(data=df,x='movement',color=base_color)
```

```
plt.ylabel("Movement counts");
plt.title('Number of transactions by Movement');
```

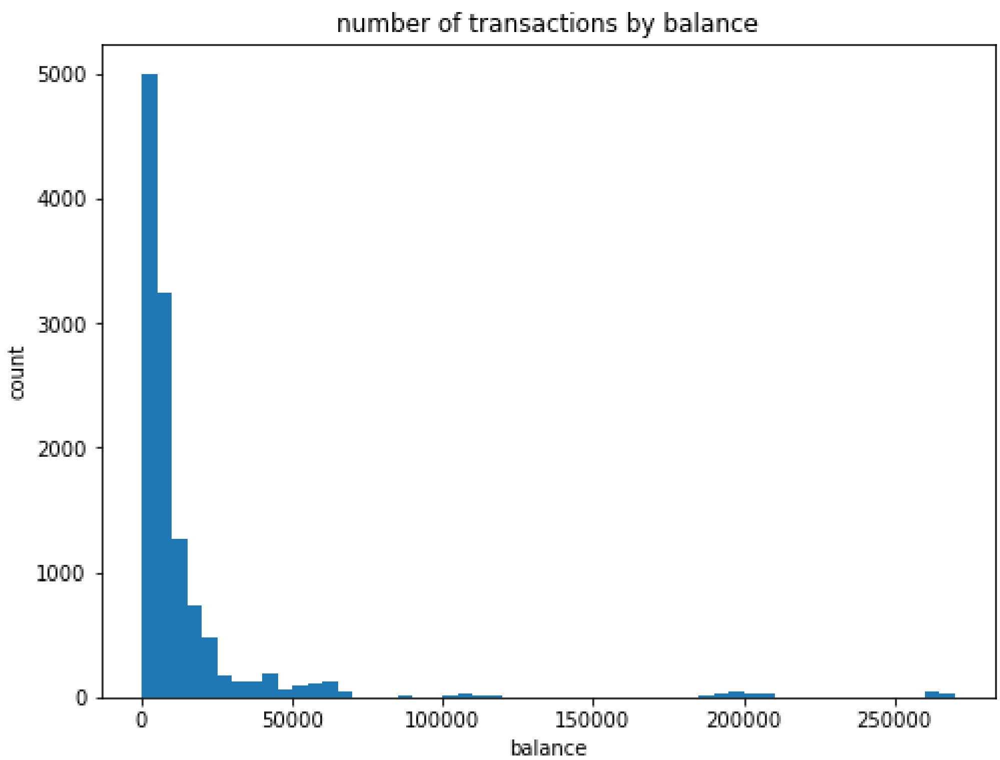


The most transactions are debit, just a little transaction are credit.

Numerical variables

In [22]:

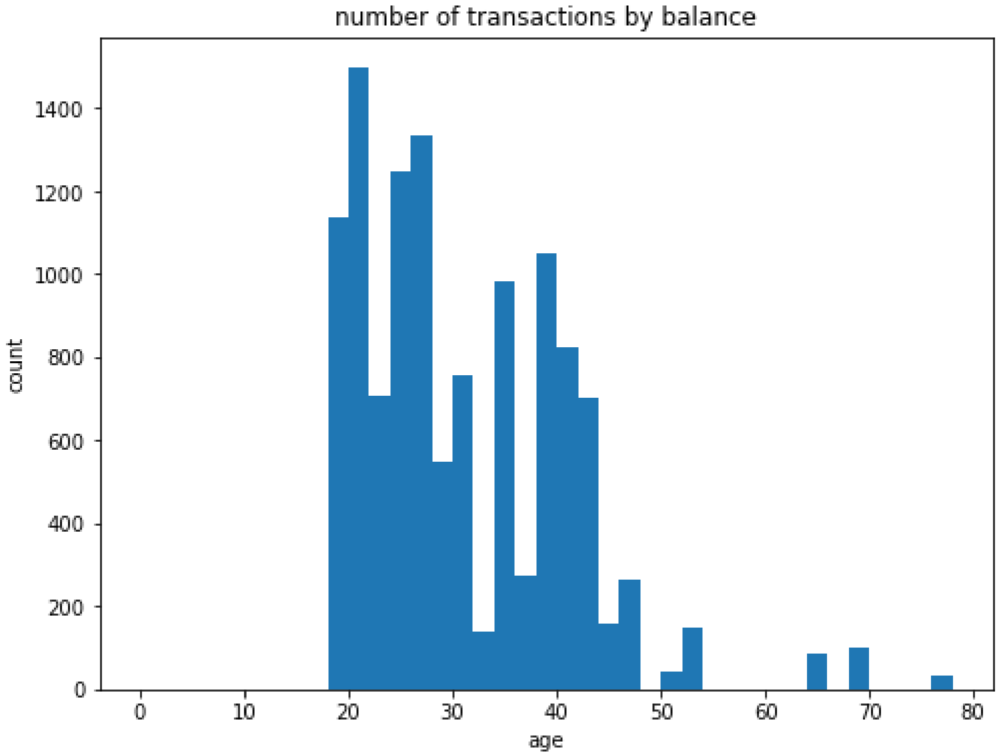
```
#Balance
plt.figure(figsize=[8,6])
default_color = sns.color_palette()[0]
plt.title('number of transactions by balance')
binsize=5000
bins = np.arange(0, df['balance'].max()+binsize, binsize)
plt.hist(data =df,x ='balance', bins = bins,color=default_color)
plt.xlabel('balance')
plt.ylabel('count')
plt.show();
```



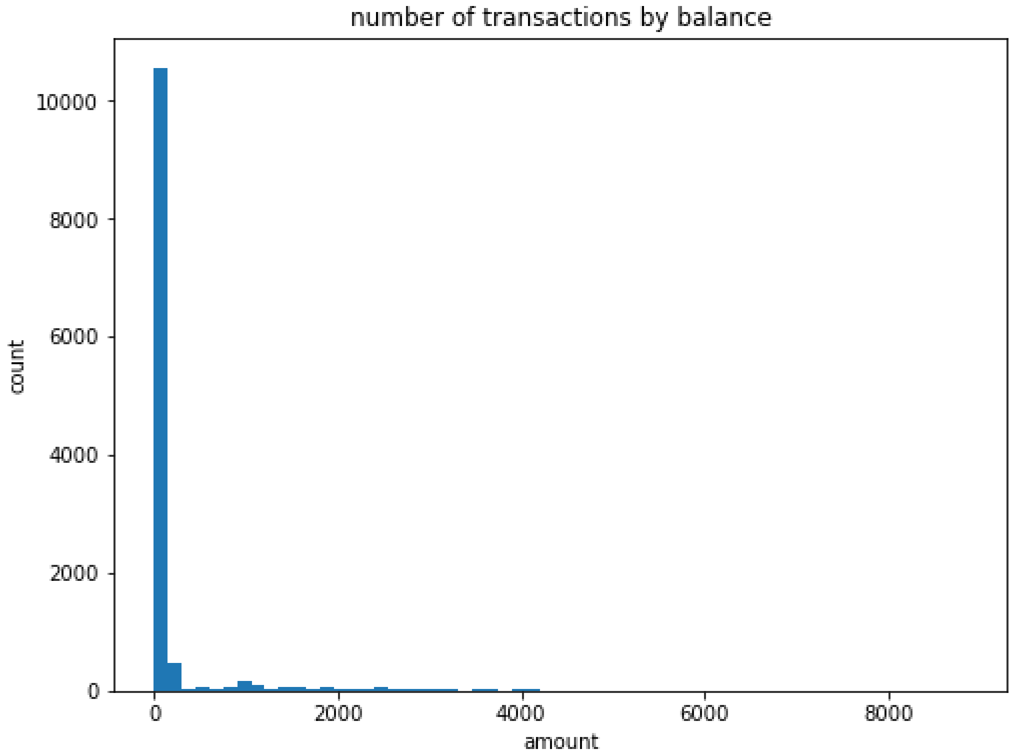
In [23]:

```
#age
plt.figure(figsize=[8,6])
default_color = sns.color_palette()[0]
```

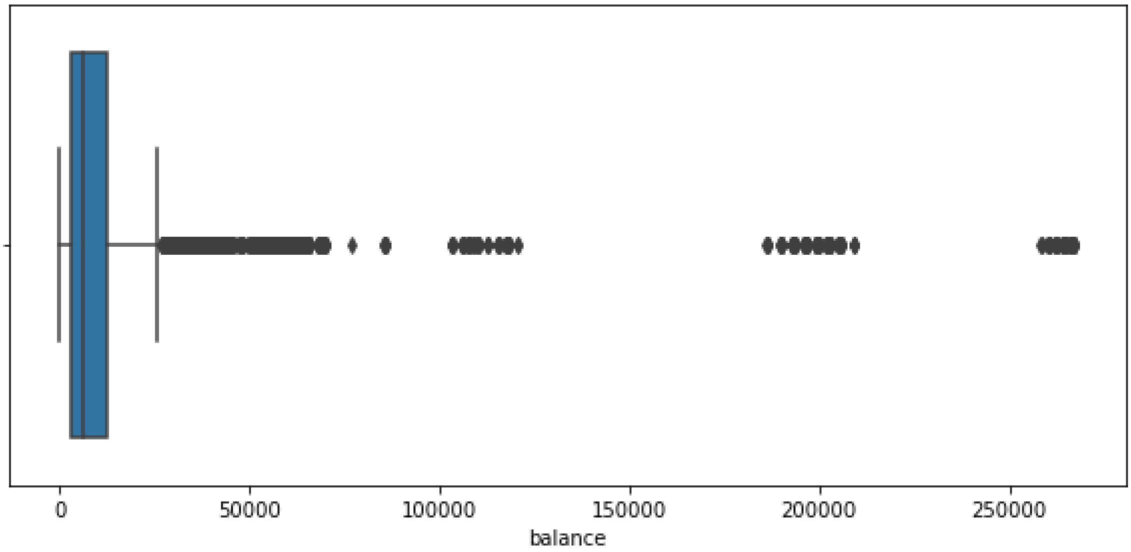
```
plt.title('number of transactions by balance')
binsize=2
bins = np.arange(0, df['age'].max()+binsize, binsize)
plt.hist(data =df,x ='age', bins = bins,color=default_color)
plt.xlabel('age')
plt.ylabel('count')
plt.show();
```

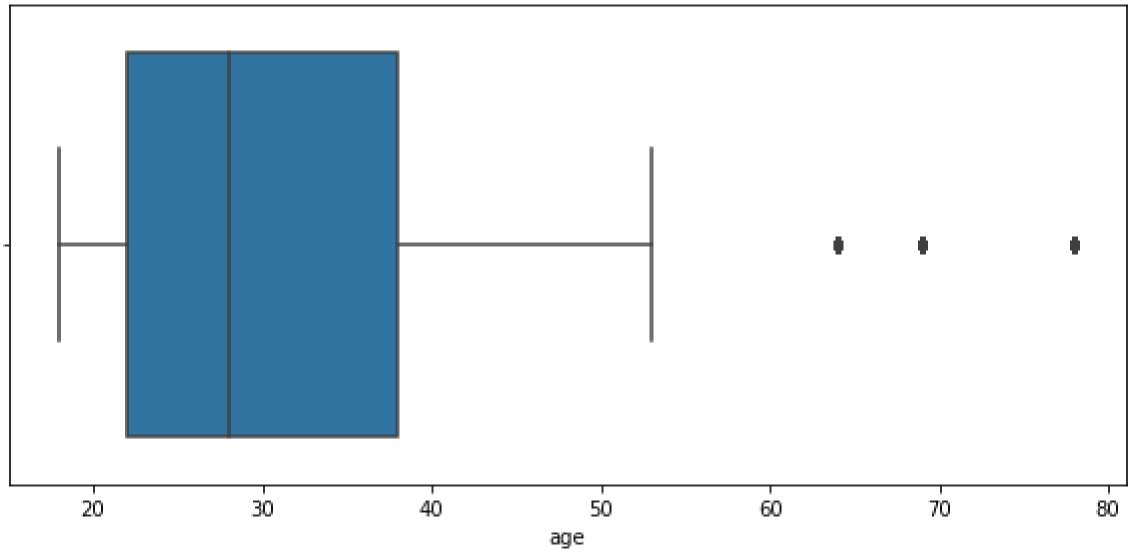


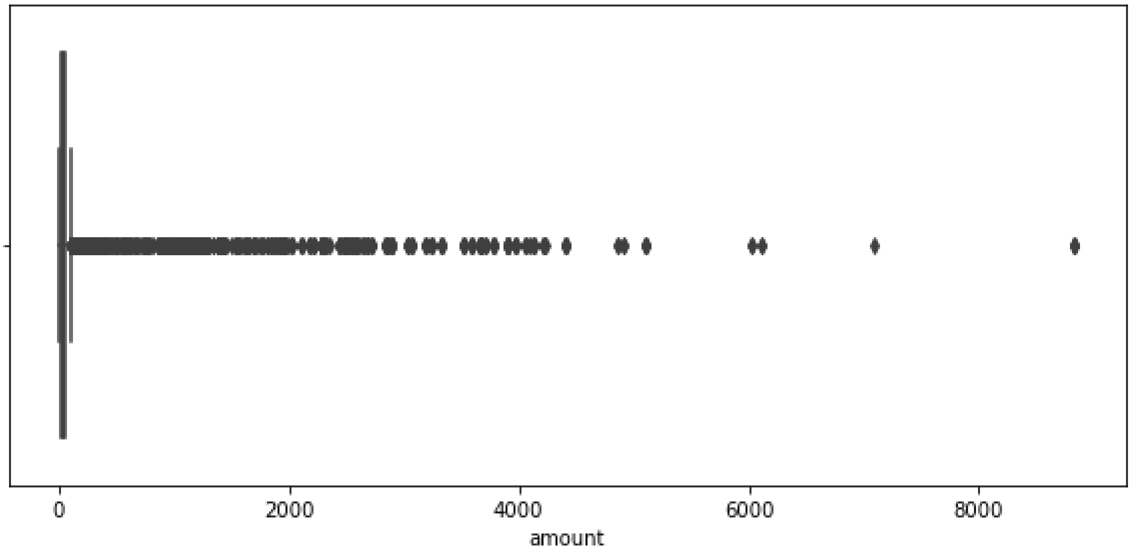
In [24]:
```
plt.figure(figsize=[8,6])
default_color = sns.color_palette()[0]
plt.title('number of transactions by balance')
binsize=150
bins = np.arange(0, df['amount'].max()+binsize, binsize)
plt.hist(data =df,x ='amount', bins = bins,color=default_color)
plt.xlabel('amount')
plt.ylabel('count')
plt.show();
```



In [25]:
```
fig,axs=plt.subplots(nrows=3,figsize=(10,15))
sns.boxplot(df['balance'],ax=axs[0])
sns.boxplot(df['age'],ax=axs[1])
sns.boxplot(df['amount'],ax=axs[2])
for ax in axs:
    ax.ticklabel_format(style='plain',axis='x')
plt.show()
```







The distribution of balance and the amount has a long tail, for the distribution of age, it seems to be the normal distribution and only have a few outliers.

Data Cleaning

Drop the columns of currency,country,merchant code and bpay_biller_code since they only have one value in the column, not provide useful information

In [26]:
```
df_clean=df.copy()
```

```
print(df.shape)
df_clean=df.drop(['currency','country','merchant_code','bpay_biller_code'],ax
print(df_clean.shape)
```

```
(12043, 23)
(12043, 19)
```

Deal with the missing data: card_present_flag merchant_id merchant_suburb merchant_state merchant_long_lat These column have the same numbers of missing value and all of them are about merchant,

In [27]:

```
cols = ["card_present_flag", "merchant_state", "merchant_suburb", "merchant_i
for col in cols:
    df_clean[col].fillna( 'null',inplace = True)
```

In [28]:

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12043 entries, 0 to 12042
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   status             12043 non-null  object
 1   card_present_flag  12043 non-null  object
 2   account            12043 non-null  object
 3   long_lat           12043 non-null  object
 4   txn_description    12043 non-null  object
 5   merchant_id        12043 non-null  object
 6   first_name         12043 non-null  object
 7   balance            12043 non-null  float64
 8   date               12043 non-null  datetime64[ns]
 9   gender             12043 non-null  object
 10  age                12043 non-null  int64
 11  merchant_suburb    12043 non-null  object
 12  merchant_state     12043 non-null  object
 13  extraction         12043 non-null  object
 14  amount             12043 non-null  float64
 15  transaction_id     12043 non-null  object
 16  customer_id        12043 non-null  object
 17  merchant_long_lat  12043 non-null  object
 18  movement           12043 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(15)
memory usage: 1.7+ MB
```

In [29]:

```
df_clean.isnull().sum()
```

Out[29]:

```
status               0
card_present_flag    0
account              0
long_lat             0
txn_description      0
merchant_id          0
first_name           0
balance              0
date                 0
gender               0
age                  0
merchant_suburb      0
merchant_state       0
extraction           0
amount               0
transaction_id       0
customer_id          0
merchant_long_lat    0
movement             0
dtype: int64
```

Feature Engineering

Split the time from the extration

In [30]:
```
df_clean["extraction"] = [timestamp.split("T")[1].split(".")[0] for timestamp
df_clean['extraction'].head()
```

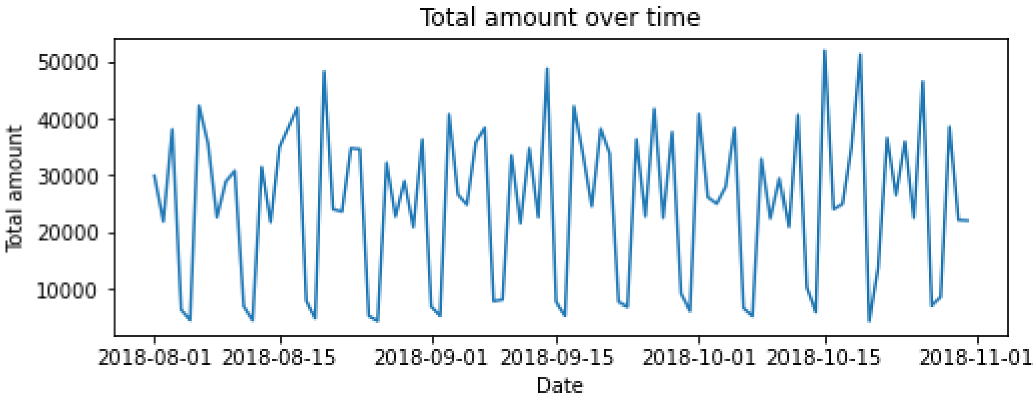
Out[30]:
```
0    01:01:15
1    01:13:45
2    01:26:15
3    01:38:45
4    01:51:15
Name: extraction, dtype: object
```

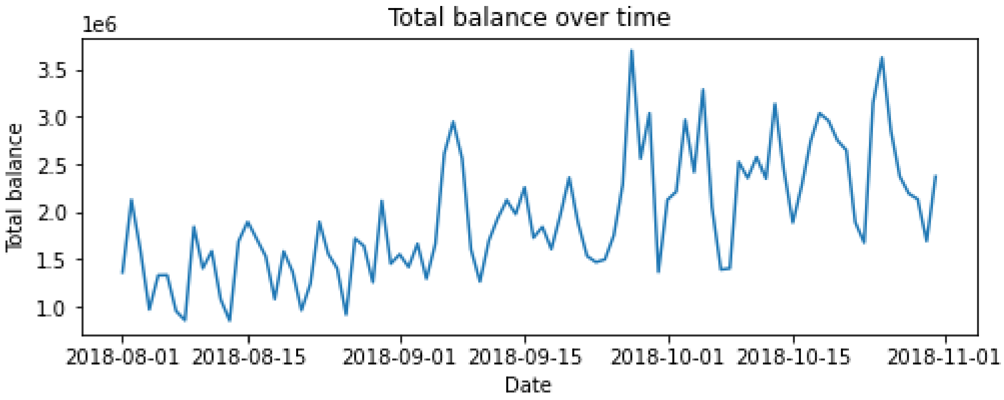
Segement Data and Visulization

Creat new dataframe which contains total amount and balance for each day

In [31]:
```
day_amount=pd.pivot_table(df_clean,values ='amount', index ='date', aggfunc =
day_balance=pd.pivot_table(df_clean,values ='balance', index ='date', aggfunc
```

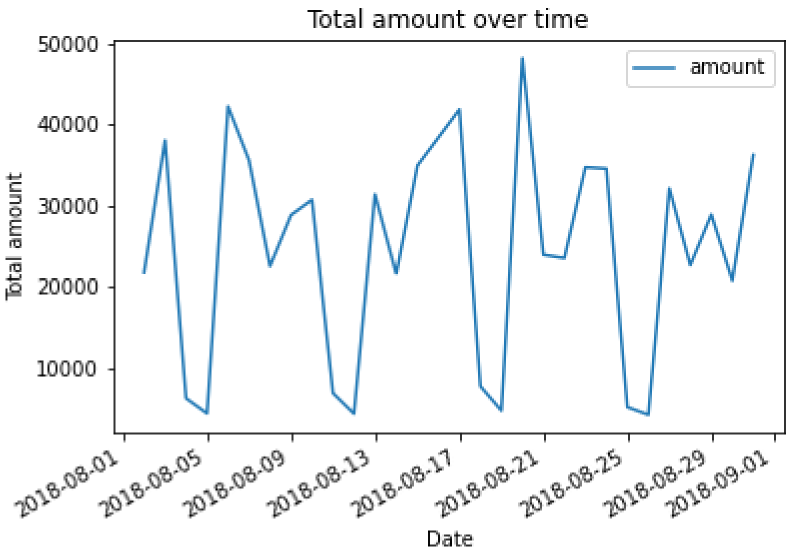
In [32]:
```
#This time I will findout the relationship between these numeric varibles and
plt.figure(figsize = [8,6])

ax = plt.subplot(2, 1, 1)
sns.lineplot(data=day_amount, x ='date',y='amount',color=default_color)
plt.title('Total amount over time')
plt.xlabel('Date')
plt.ylabel('Total amount')
plt.show()

plt.figure(figsize = [8,6])
ax = plt.subplot(2, 1, 2)
sns.lineplot(data=day_balance, x ='date',y='balance',color=default_color)
plt.title('Total balance over time')
plt.xlabel('Date')
plt.ylabel('Total balance')
plt.show()
```





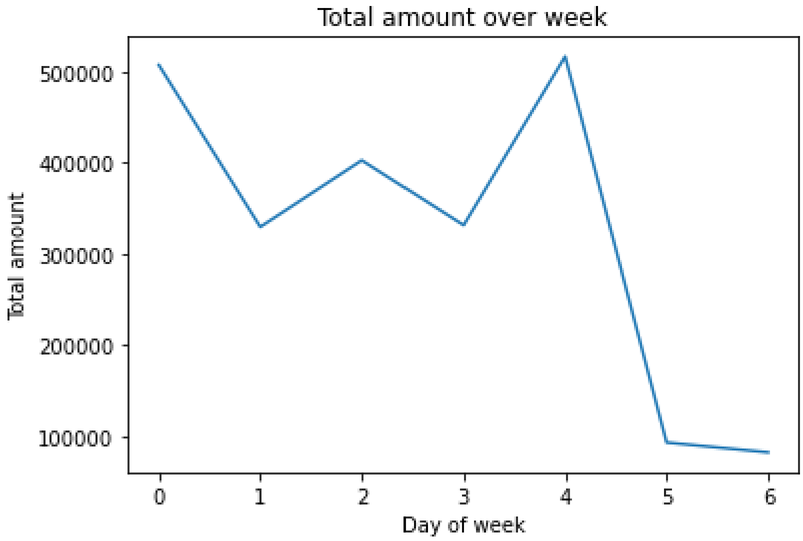
The balance seems continosly increase which is make sense, but the amount have some regular pattern over the 3 month. We will extract one month data and the day of week to analysis the amount over time.

In [33]:
```
#See the transaction in August
Aug_date_total = day_amount[(day_amount.index > "2018-8-1") & (day_amount.ind
Aug_date_total.plot(kind='line',figsize=(6,4))
plt.title('Total amount over time')
plt.xlabel('Date')
plt.ylabel('Total amount')
plt.show()
```



In [34]:
```
df_clean["dayofweek"] = pd.DatetimeIndex(df_clean.date).dayofweek
week_amount=pd.pivot_table(df_clean,values ='amount', index ='dayofweek', agg
```

In [35]:
```
sns.lineplot(data=week_amount, x ='dayofweek',y='amount',color=default_color)
plt.title('Total amount over week')
plt.xlabel('Day of week')
plt.ylabel('Total amount')
plt.show();
```



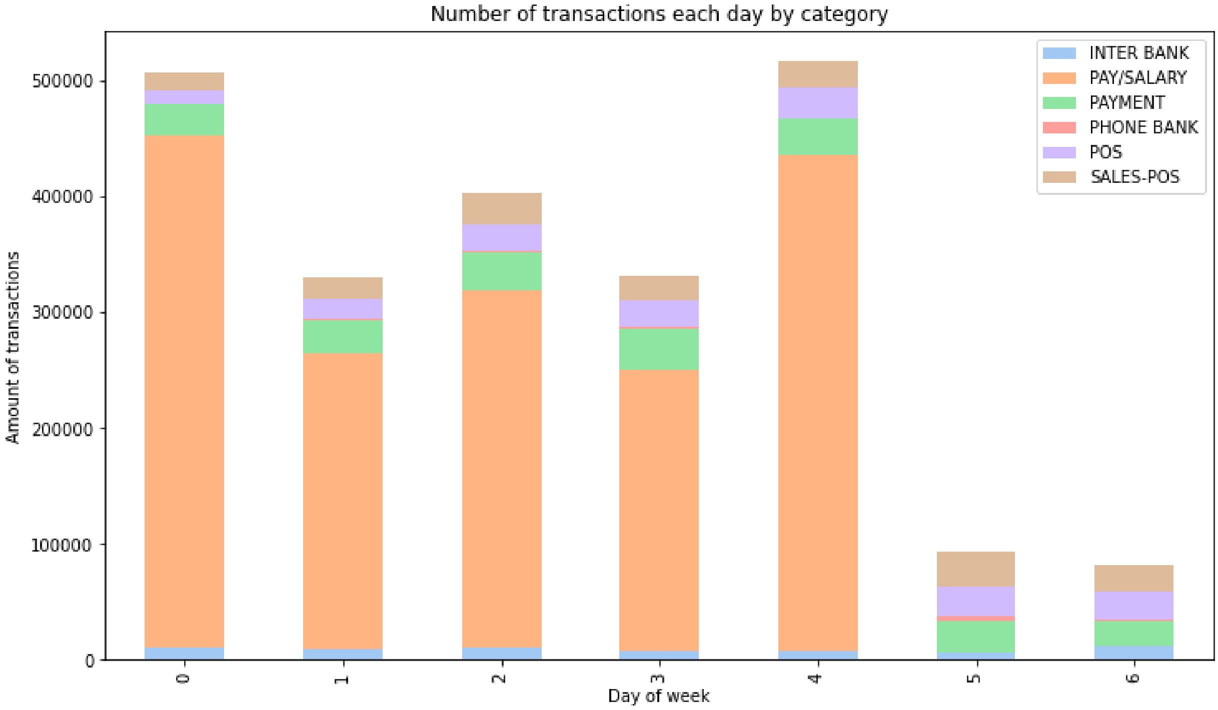
The total transaction amount are the lowest in Friday and Saturday, as for the reason, we can breakdown the transaction types

In [36]:
```
df_clean.groupby(['txn_description'],as_index=False)['amount'].sum().sort_val
```

Out[36]:

	txn_description	amount
3	PHONE BANK	10716.00
0	INTER BANK	64331.00
4	POS	152861.24
5	SALES-POS	157005.11
2	PAYMENT	201794.00
1	PAY/SALARY	1676576.85

In [37]:
```
stacked_barplot = pd.DataFrame(df_clean.groupby(["dayofweek", "txn_descriptio
default_color=sns.color_palette('pastel')
stacked_barplot.unstack().plot(kind = "bar", stacked = True, figsize = (12, 7
plt.title("Number of transactions each day by category")
plt.legend(["INTER BANK", "PAY/SALARY", "PAYMENT","PHONE BANK","POS",'SALES-P
plt.ylabel("Amount of transactions")
plt.xlabel("Day of week");
```



There is no salaries were paid on Friday and Saturday, and the lowest amount is phone bank.

Creat new dataframe which contains total amount and balance for each day

In [48]:

```
df_clean["hour"] =[time.split(":")[0] for time in df_clean['extraction']]
```

```
  File "<ipython-input-48-461fb28f25f3>", line 1
    df_clean["hour"] = df_clean[time.split(":")[0] for time in df_clean['extra
ction']]
                                                    ^
SyntaxError: invalid syntax
```

In [49]:

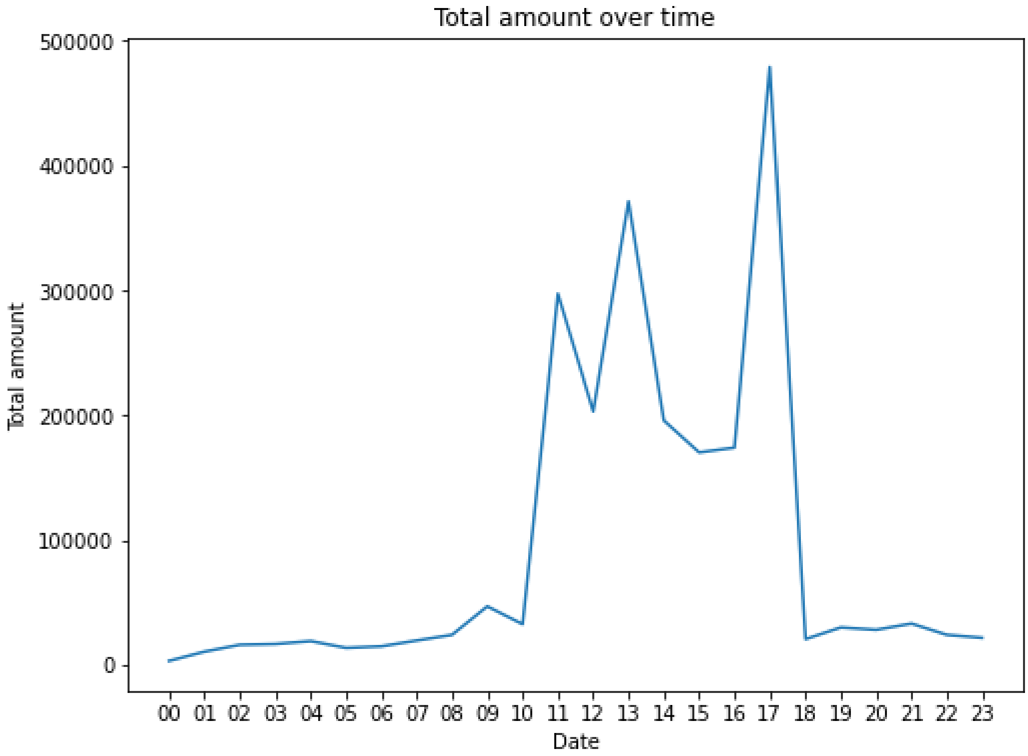
```
df_clean["hour"]=df_clean['extraction'].str[:2]
```

In [50]:

```
hour_amount=pd.pivot_table(df_clean,values ='amount', index ='hour', aggfunc =
```

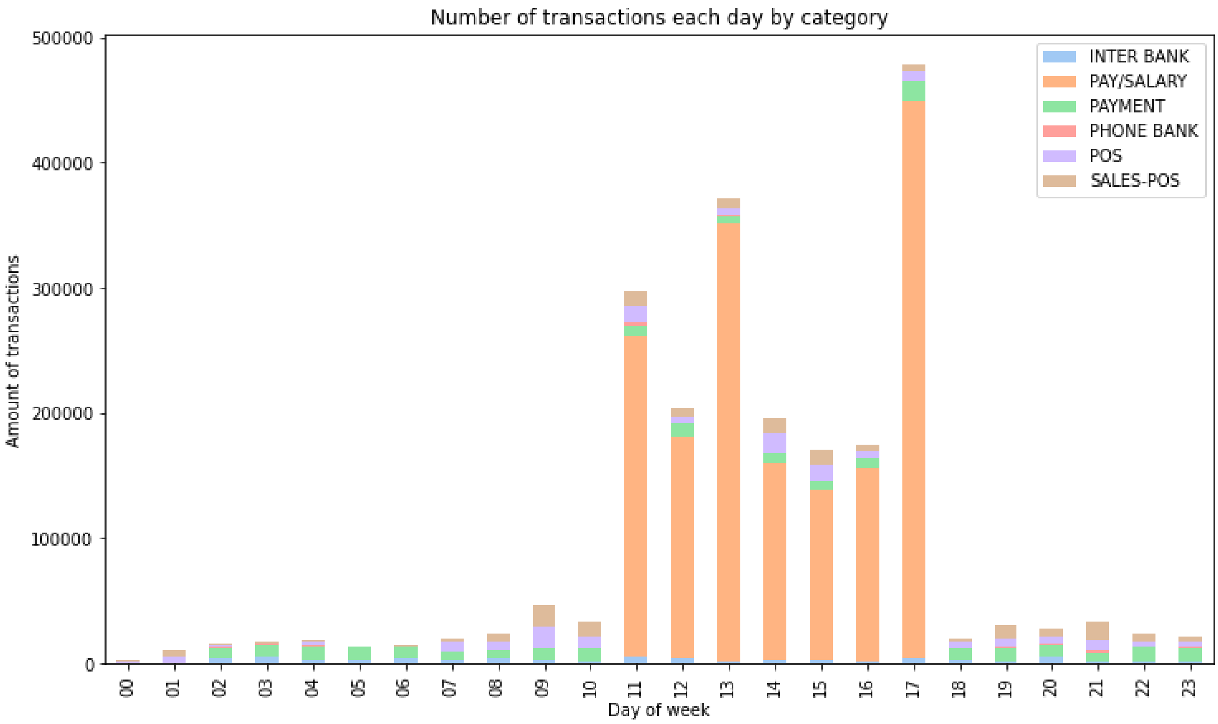
In [57]:

```
fig,ax=plt.subplots(figsize = (8,6))
ax.plot(hour_amount.index, hour_amount.amount)
plt.title('Total amount over time')
plt.xlabel('Date')
plt.ylabel('Total amount')
plt.show()
```



Most transactions are happend between noon and afternoon

In [41]:
```
stacked_barplot = pd.DataFrame(df_clean.groupby(["hour", "txn_description"]).
stacked_barplot.unstack().plot(kind = "bar", stacked = True, figsize = (12, 7
plt.title("Number of transactions each day by category")
plt.legend(["INTER BANK", "PAY/SALARY", "PAYMENT","PHONE BANK","POS",'SALES-P
plt.ylabel("Amount of transactions")
plt.xlabel("Day of week");
```



Visualize the location data

First we need to separate the longtitude and latitude

In [42]:
```
df_clean["longtitude"] = [string.split(" ")[0] for string in df_clean['long_l
df_clean["latitude"] = [string.split(" ")[1] for string in df_clean['long_lat
```

```
print('longtitude:',df_clean["longtitude"].head(1))
print('longtitude:',df_clean["latitude"].head(1))
```

```
longtitude: 0    153.41
Name: longtitude, dtype: object
longtitude: 0    -27.95
Name: latitude, dtype: object
```

In [43]:

```
df_clean["merchant_longtitude"]=df_clean['merchant_long_lat'].str[:6]
df_clean['merchant_latitude']=df_clean['merchant_long_lat'].str[7:]
print('merchant_longtitude:',df_clean["merchant_longtitude"].head(1))
print('merchant_longtitude:',df_clean["merchant_latitude"].head(1))
```

```
merchant_longtitude: 0    153.38
Name: merchant_longtitude, dtype: object
merchant_longtitude: 0    -27.99
Name: merchant_latitude, dtype: object
```

Create the map of the data

In [66]:

```
import folium
```

In [79]:

```
#Create a map of Australia
latitude=27.00
longitude=133.00
aus_map = folium.Map(location=[latitude, longitude], zoom_start=5)
aus_map
```

Out[79]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [80]:

```
transactions = folium.map.FeatureGroup()
for lat, lng, in zip(df_clean.latitude, df_clean.longtitude):
    transactions.add_child(
        folium.CircleMarker(
            [lat, lng],
            radius=5,
            color='yellow',
            fill=True,
            fill_color='red',
            fill_opacity=0.4
        )
    )
aus_map = folium.Map(location=[latitude, longitude], zoom_start=5)
aus_map.add_child(transactions)
```

Out[80]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [81]:
```
from folium import plugins

auc_map = folium.Map(location = [latitude, longitude], zoom_start =5)
incidents = plugins.MarkerCluster().add_to(auc_map)

for lat, lng, label, in zip(df_clean.latitude, df_clean.longtitude, df_clean.
    folium.Marker(
        location=[lat, lng],
        icon=None,
        popup=label,
    ).add_to(transactions)

# add incidents to map
san_map.add_child(transactions)
```

Out[81]: Make this Notebook Trusted to load map: File -> Trust Notebook

From the map we can see that most transactions are near the sea,maybe the office location are near sea, for more visualization I will make a dashborad by tableau.